Argumentative Design Rationale and the Object-Oriented Development Process: better analysis, better design

LeeRoy Bronner, Ph.D., P.E., Jerry-Daryl Fletcher

Morgan State University

Introduction

Many analysis and modeling problems done today for information technology applications lead to the solution of system problems. In the development of these solutions, reasoning is a major component. The reasoning component which is normally neglected can be captured in Rationale Models. Rationale Models represent the reasoning that lead to the system solution. This reasoning is defined as Design Rational (DR). There have been a number of research studies into DR, however, in this research, it was found that industry has neglected DR in their system analysis because of the increased time and effort required to capture and implement DR. Some of the benefits of DR are: 1) maintenance is more efficient and effective, 2) system scalability is increased, and 3) training of users and developers is easier. This paper proposes a systematic approach to the capture of argumentative DR and an integration of argumentative DR with the Object-Oriented system development lifecycle. Change is a constant in the implementation and use of systems, hence, this paper also raises the issue of “how should argumentative DR be stored and integrated with the system to maximize its utility to the system.

What is Rationale?

Rationale is an explanation of controlling principles of opinion, belief, practice, or phenomena; or an underlying reason or basis. From these two definitions we gather that rationale is the actual brain work that is done behind everything that is done. When we are encountered with a new problem or phenomenon, after we make our observations, we rationalize the problem/situation in order to formulate a hypothesis, use these hypotheses to predict the existence of other phenomenon and then conduct experiments based on our hypotheses and then we draw final conclusions, according to the scientific research method [13]. Thus rationale is the intellectual foundation on which systems are made. For the purposes of this paper, a system, is defined as a regularly interacting or interdependent group of items forming a unified whole. Thus when the word system is used, it not only refers to a software system, but to any activities that fit the aforementioned definition.

What is DR?

Design rationale (DR) is the reasoning that goes into determining the design of the artifact. It can include not only direct discussion of artifact properties but also any other reasoning influencing design of the artifact [7]. DR can be characterized by the approaches that are taken to it, namely descriptive or prescriptive, and intrusiveness into the design process [7].

Descriptive approaches to rationale refer to processes in which the goal of the DR is to describe the thinking process that the system designer(s) utilize. Alternatively, prescriptive approaches are aimed at improving the design process by improving the reasoning process of the system designers [1]. Also, the extent to which the method of DR capture intrudes in the design process is a characterizing feature. Most of the DR approaches are of the intrusive nature, though over the past 15 years there has been extensive research done to find less intrusive ways of capturing and formalizing DR [7]. This work in reducing the intrusiveness of DR is being done in an attempt reduce the overhead involved with capturing and utilizing DR, and make it much more intuitive to designers.

What are the benefits of Design Rationale?

Because of the constant change in virtually every industry, any system not built from solid rationale is at a disadvantage to systems built with solid rationale. By effectively and efficiently capturing and integrating the rationale of a system with that system, the overall quality of the system is increased. This increased system quality is evidenced by improved system maintenance, scalability, training, reuse and documentation. Dutoit et al. lists collaboration support, reuse and change support, quality improvement and knowledge transfer support as the four broad areas of utilization for DR and DR capture methods.

What is the state of the art of DR in the industry?

There are many tools that are currently used to capture DR. The latest and most widely utilized tool is Compendium [12], which is an open-source tool with an international developer and user community. It is essentially an extension of IBIS and gIBIS, and QuestMap, an older tool. Compendium offers an ever-expanding set of functionalities, and looks destined to be the tool of choice for the future. According to Buckingham Shum et al. Compendium is understood when analyzed through the following three dimensions: (1) its functionality for hypermedia concept mapping, (2) how it uses IBIS to support collaborative modeling of a problem using any conceptual framework, and (3) in the context of mapping ideas in real-time during a meeting [3].

Argumentative DR

Argumentative DR is the expression of DR as largely, semi-formal arguments. The notion that DR should be represented as semi-formal arguments can be traced back to wider research into the development of computational support for reasoning [1], [2]. There are several approaches to argumentative DR, namely, Issue Bases Information System (IBIS), Graphical IBIS (gIBIS), Procedural hierarchy of issues (PHI), Decision Representation Language (DRL) and Questions, Options, and Criteria (QOC).

IBIS, developed in the 1970's by Horst Rittel as a medium to address wicked problems. Wicked problems, according to Rittel have the following characteristics [5]:

- ***You don't understand the problem until you have developed a solution***: with each proposed solution raising previously unforeseen issues to be addressed
- ***There is no stopping rule***: lacking a definite problem, there is also no definite solution
- ***There is no absolute solution***: solutions are all relative; better or worse, but never right or wrong
- ***Every wicked problem is unique***: hence the repository for a solution set for wicked problems is rendered useless
- ***Every solution to a wicked problem is a "one-shot solution"***: in order to learn about the problem you must try a proposed solution, which incurs cost
- ***There are no given alternatives***: the discovery and choice of alternatives is based on the imagination and creativity of the designer, thus some alternatives are never even thought of.

The key elements of IBIS are Issues, Positions and Arguments. The Issues (question/problem) are given proposed Positions (alternatives) that are evaluated based on Arguments about their relative strengths and weaknesses to each other. As Positions are explored, more Issues arise, and hence the IBIS network is expanded to include this new Issue and its associated Positions and Arguments. There are nine kinds of links in IBIS. For example, a Position Responds-to an Issue, and this is the only place the Responds-to link can be used. Arguments must be linked to their Positions with either Supports or Objects-to links. Issues may Generalize or Specialize other Issues, and may also Question or Be-suggested-by other Issues, Positions, and Arguments [4] (Figure 1).

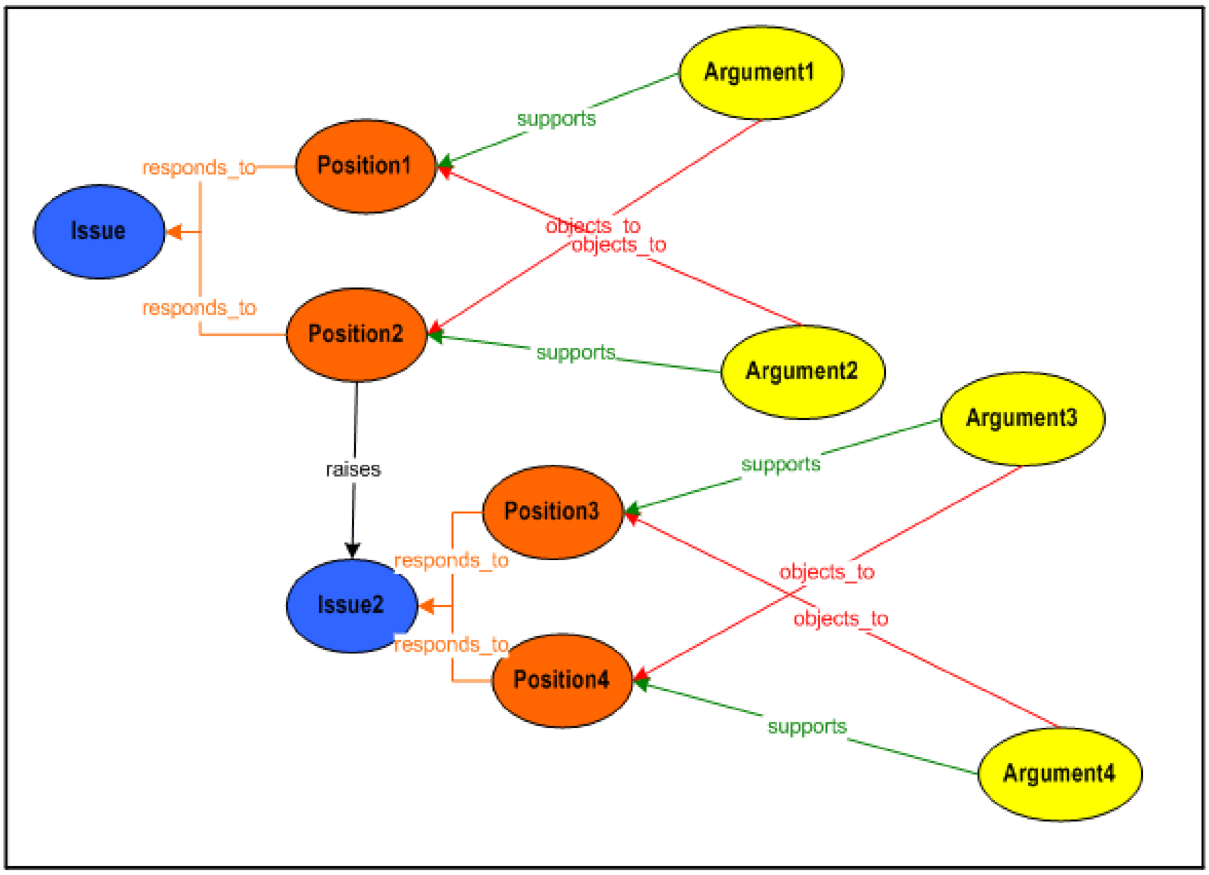

Figure 1 showing the standard IBIS model with issue, position and argument nodes, and possible relationships between them.

gIBIS is a hypertext tool that was developed by Jeff Conklin and Michael Begeman to support the capture of DR, by applying IBIS [4].

PHI was developed by Raymond McCall as an extension of the IBIS ideology. The key to PHI is the improving of design reasoning by raising subissues [7]; developing on the premise that exploration of these relationships would strengthen design. PHI differs from IBIS in two respects: it allows decomposition of issues, answers and arguments into hierarchies of subissues, subanswers and subarguments; and it broadens the concept of *issue to* include all design questions, not merely those deliberated [8].

DRL is essentially an extension of IBIS which has the capability to provide a finer level of granularity than IBIS. It does not cover all aspects of design rationale, but rather focuses on decision rationale, providing guidance on the generation of design alternatives [7]. The base elements in DRL are Decision problems, Alternatives, Goals, Claims and Groups. The relationships between these elements are very similar to those in IBIS, and are shown in (Figure 2) [1].

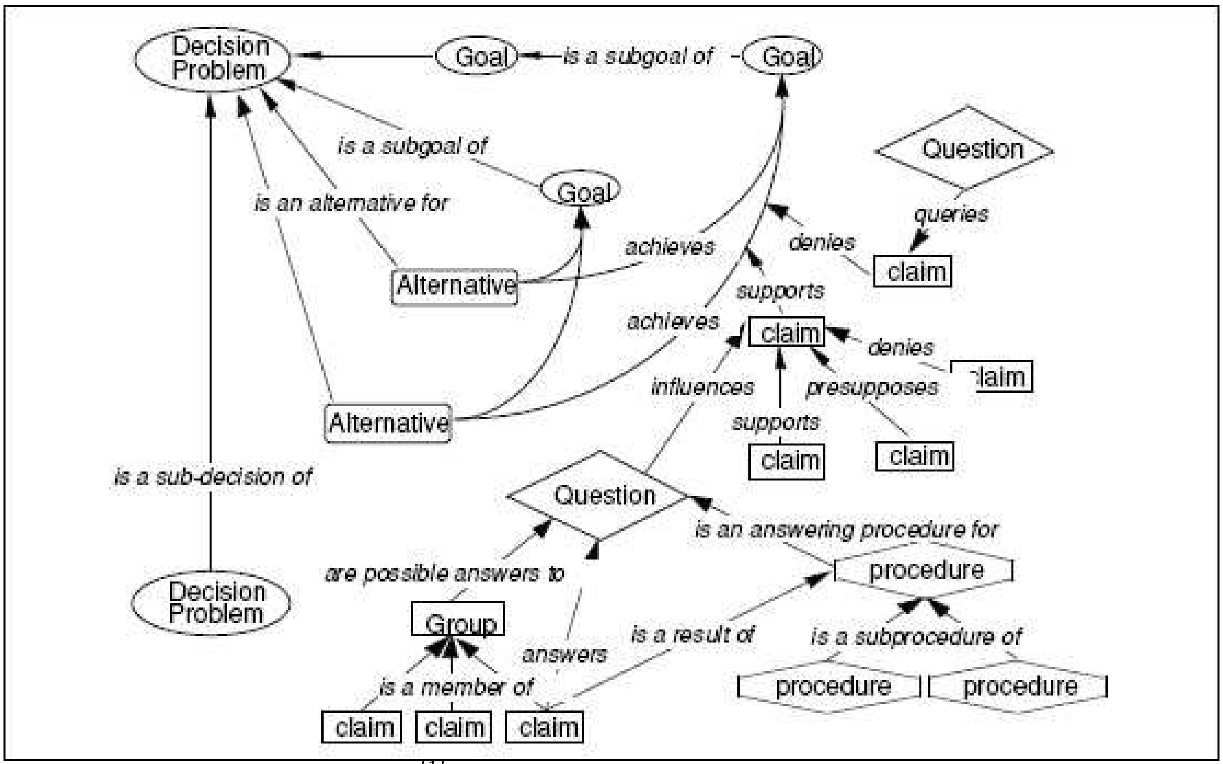

Figure 2 showing DRL structure [1].

QOC [9] comprises six major elements; Questions, Options, Criteria, Assessments, Arguments and Decisions. The relationships between these elements are similar to the relationships used in IBIS, and are shown in (Figure 3). However, unlike in IBIS where questions can deal with any design topic, QOC's questions deal exclusively with features of the artifact being designed [7].

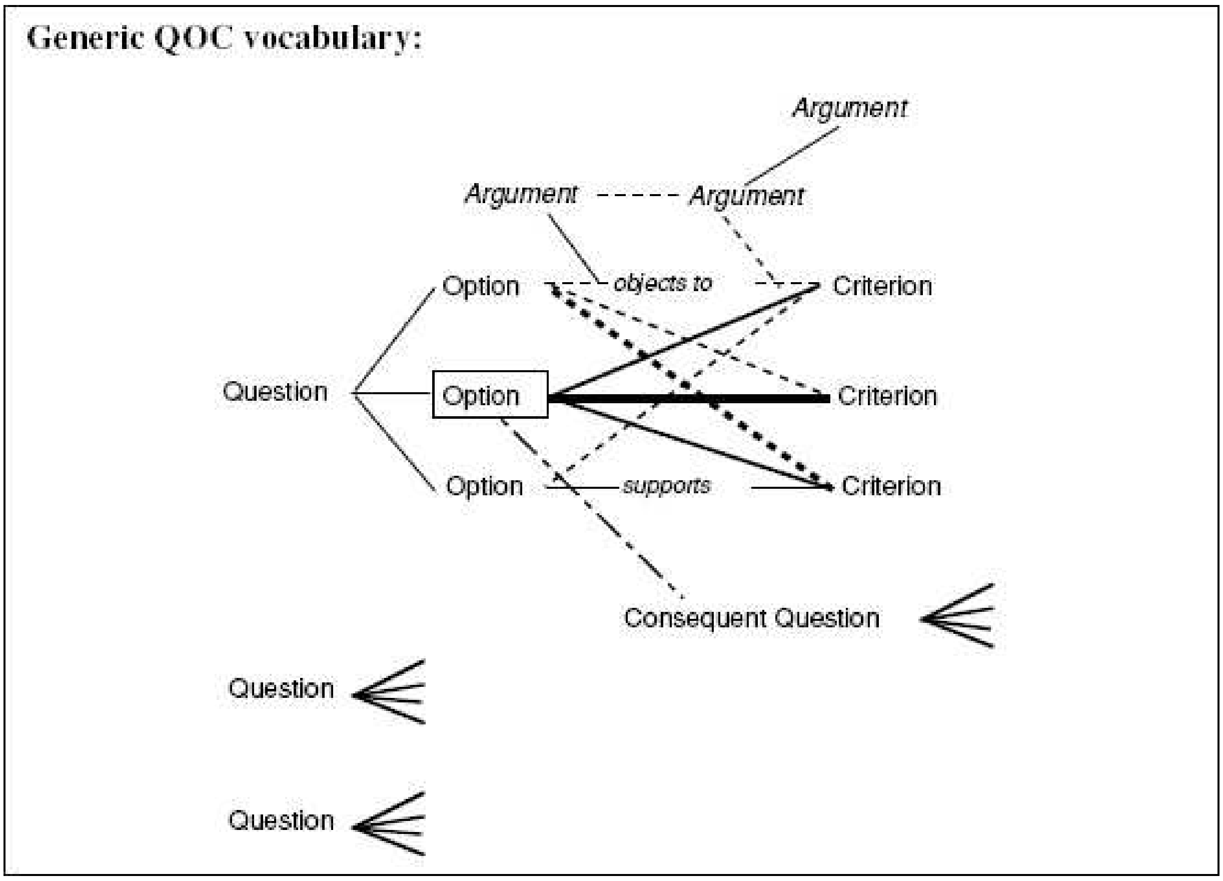

Figure 3 showing the generic QOC structure, with line thickness denoting relative weights of assessments [2].

DR Capture Issues

Even with the advent of tools such as Compendium, and research dating back over a quarter century, there has been reluctance by many in industry to utilize DR. This underutilization is due mainly to the problems associated with capturing DR. According to Dutoit et al. [7], some of the problems associated with DR capture can be attributed to intrusiveness, political and legal issues, fundamental problems with descriptive approaches, time and cognitive overhead.

The fact that most approaches to DR utilize their own schema, the intrusiveness of DR capture is high, since system designers must not only design the system, they must also adhere to a given DR schema. Also, designers do not always prefer to disclose their real reasons for design, especially when it deals with political issues. In descriptive approaches, the beneficiaries of the DR are not the designers themselves, and this leads to disinterest in DR capture [7]. In industry, one of the most important resources is time, and by adding DR capture as part of the system process, more time is required by designers. This leads to additional costs to offset this time, and makes DR unattractive not just to designers, but to management as well. There is a cognitive overhead associated with DR capture which not only slows down the physical output of designers on design, but also slows down their mental output on design as well [10].

When and where is it necessary to capture DR?

In a perfect system setting, it would be possible and practical to capture the entire DR for all the design decisions made in a system. However, in reality it is highly impossible and impractical to attempt to capture a system's entire DR. Considering that several options are often considered before any one decision is made; the potential size of the system DR would astronomically supersede the size of the system itself. Then consider a system as complex as Windows XP® that involved hundreds of developers over a long period of time; the DR would become an impenetrable fortress that would prevent the completion of the system. This is the concern of many developers who do not perceive the practicality in applying DR because of the complex nature of the systems that they construct, thus the conundrum, when should or should not DR be captured?

A careful inspection of the system being developed is required in order to answer this question. According to Dalrymple, the rationale model should be created for the major decisions [6]. However, what are the criteria for determining whether or not a decision should be deemed as major for a system? The determination of major/critical decisions should be an iterative process that begins in the conceptualization stage of a system. The starting point for DR capture should be the motivations for the system itself, without looking at the components of the system. An understanding of exactly why the system is being developed, what environment it is being developed in, its perceived benefits and drawbacks, and its intended users should be captured.

DR Capture Theory in the Object-Oriented Paradigm

By utilizing the Object-Oriented (OO) paradigm (Figure 4), the DR capture can actually begin at the initial problem definition stage. During use case analysis the system is developed and expanded, and capturing argumentative DR at this point of system development will promote healthy discussion between developers and clients during the Joint Application Design (JAD) sessions. After the argumentative DR is used and captured to fully describe the system through use case analysis, the use case analysis is used to drive other system analysis processes.

The major goals of all system analysis processes are system definition and requirements elicitation and analysis. The requirements of a system describe the conditions or capabilities to which a system must conform, and represent what the system should do, as opposed to how it should be built [11]. The design of the system is done after the requirements elicitation and analysis and is driven by their output. The analysis phase is the most essential phase in the system since it defines the entire system. As such, the argumentative DR of the analysis stage can be used as the over-arching system DR, since at each other phase of system development; the decisions must positively support the system requirements and specifications.

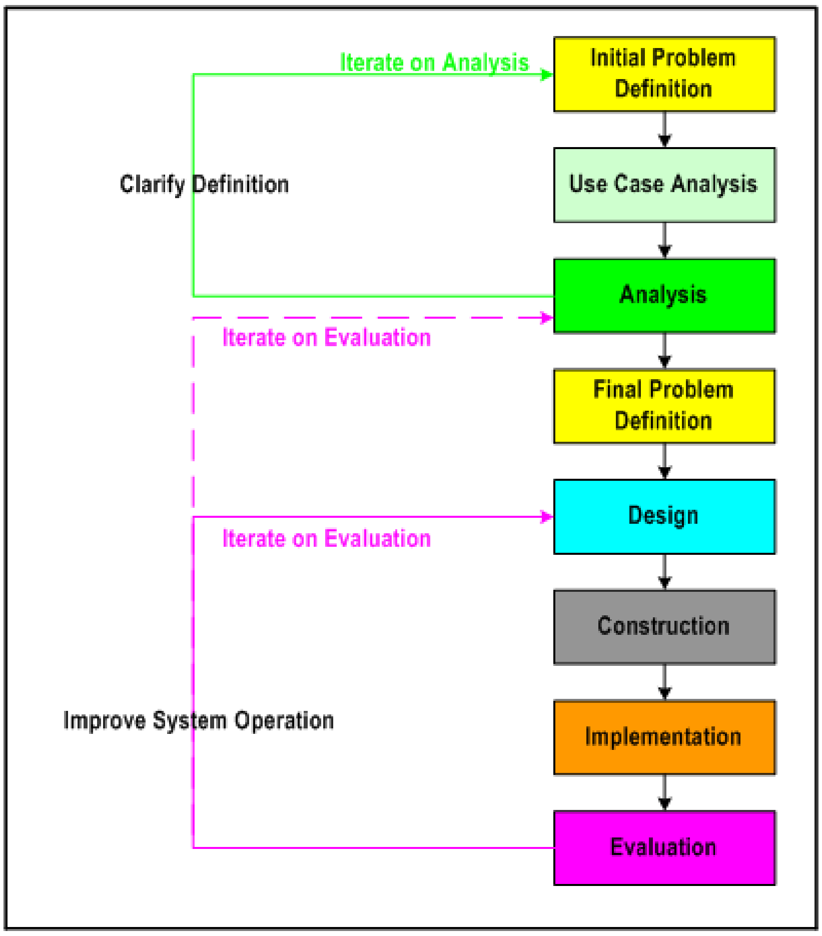

Figure 4 showing the Object-Oriented Development Lifecycle.

During design, argumentative DR should be captured for design decisions that oppose or cause modification to the DR captured at the analysis phase (Figure 5). These opposing decisions form the iterative loops that create more complete correct systems. Intuitively, when we find something that challenges the modus operandi and accepted norms, we revisit the modus operandi and accepted norms to ensure their correctness. If they remain correct then we discard the new notion, but if the new notion proves to be correct, then we modify the norms to include it. In like manner DR can be measured against the system rationale captured at the analysis phase to ensure that the system remains complete and correct.

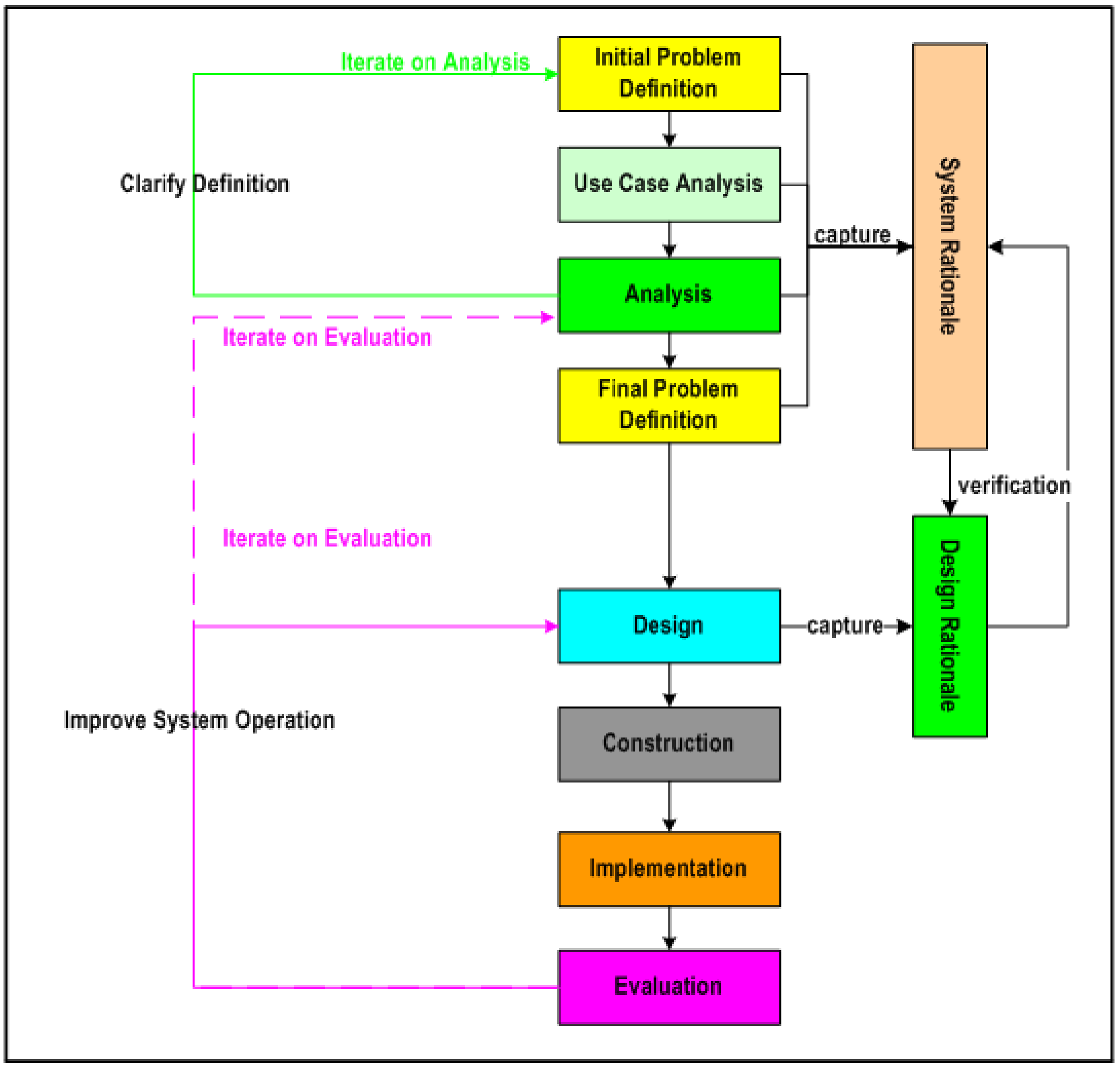

Figure 5 showing the Object-Oriented Lifecycle with DR capture integrated.

How should the DR be captured?

“The capture problem is the spectre haunting all design rationale efforts (indeed, all knowledge management efforts attempting to meaningfully capture elements of human reasoning and discourse). How does one acquire quality input to a rationale management system, without disrupting the very process it is designed to support, or without having to employ dedicated scribes who do nothing but maintain rationale libraries?”[3]

The way in which DR is captured greatly affects its utility to the system. The issues that developers have with DR revolve around finding a process that is both intuitive and efficient. If a formal process is employed, then the artifacts generated by this process will be more readily integrated with a computerized system than if a semiformal process. However, semiformal processes are much more intuitive to developers than formal processes. The answer to the formal versus semi-formal question lies in a delicate balance of the two. While there is no hard and fast answer to this question, a system-specific approach must be employed. This means that the environment, uses and users of the system must be taken into account, and then a corresponding balance of formal and semi-formal processes and notations utilized. To this end the capture of DR is a combination of art and a science.

With tools such as Compendium now allowing the capture of rationale in meetings, there is a medium for beginning to formally capture the rationale that was previously almost exclusively semi-formal at the analysis phase. Voice recorders and videotapes are now replaceable by diagrams that can be much more easily integrated with a computerized system than their predecessors. At the design phase, Compendium also allows for the capture or argumentative DR, which can then be used to reason with the analysis/system rationale.

Conclusion

By increasing the support for the analysis of a system, we are able to increase the design of the system since better analysis leads to better design. By integrating argumentative DR with the Object-Oriented Development Lifecycle for a system, the rationale of the system can theoretically be used to increase the completeness and correctness of the system. This is accomplished by capturing the rationale at the analysis phase and then using it as a measure to check the DR for consistency with the system. The approach to the argumentative DR is founded in Rittel's IBIS, and is accomplished by use of the Compendium tool.

Future Work

Though the capture of the DR has its benefits, work needs to be done in the area of the integration of this captured rationale with the system itself. Though relational databases are the most widely used throughout industry, object databases are growing in their popularity, especially when the system under study is an Object-Oriented system. To this end, by utilizing Compendium to capture the DR in an Object-Oriented system will need integration, since Compendium utilizes a relational database and the system may employ an Object-Oriented database.

Also, work needs to be done on the way that the rationale will be represented to the designers in such a way that they will not have to switch back and forth between system models and associated rationale models. This calls for an intuitive approach to integrating and representing the rationale, so that it is not a separate entity to the system development process (Figure 5), but that the system development process can more naturally resemble the regular OO development lifecycle (Figure 4), but with the rationale integrated as part of the actual phases and processes.

Bibliography

[1] Buckingham Shum S, Hammond N (1994) Argumentation-Based Design Rationale: What Use at What Cost? International Journal of Human-Computer Studies, 40 (4): 603-652

[2] Buckingham Shum S. (1996) Design Argumentation as Design Rationale. The Encyclopedia of Computer Science And Technology (Marcel Dekker Inc: NY), Vol. 35 Supp. 20, 95-128

[3] Buckingham Shum S.J, Selvin A.M, Sierhuis M, Conklin J, Haley C.B, Nuseibeh B (2005) Hypermedia Support for Argumentation-Based Rationale: 15 Years on from gIBIS and QOC. In: Dutoit A, McCall R, Mistrik I, Paech (eds.) Rationale Management in Software Engineering, Springer-Verlag, Berlin Heidelberg, Germany, pp. 111-132

[4] Conklin J, Begeman M.L. (1988) gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Trans. on Office Information Systems, 4(6), pp. 303-331

[5] Conklin J (2006) Dialogue Mapping: Building Shared Understanding of Wicked Problems. Wiley, UK, pp. 3-40

[6] Dalrymple O (2005) Using Systems Engineering Methodology to Engineer Community Based Participatory Research, Masters Thesis, Morgan State University, Dept. of Industrial Manufacturing & Information Engineering

[7] Dutoit A, McCall R, Mistrik I, Paech B (2006) Rationale Management in Software Engineering: Concepts and Techniques. In: Dutoit A, McCall R, Mistrik I, Paech (eds.) Rationale Management in Software Engineering, Springer-Verlag, Berlin Heidelberg, Germany, pp.1-48

[8] Fischer G, McCall R, Morch A (1989) Design Environments for Constructive and Argumentative Design. In: Proceedings of the SIGCHI conference on Human Factors in Computing Systems: Wings for the mind, New York, NY, US, pp. 269-275

[9] MacLean A, Young RM, Bellotti VME, Moran T (1996) Questions, Options and Criteria. In: Moran TP, Carroll JM (eds.) Design Rationale, Concepts, Techniques and Use, Lawrence Earlbaum Associates, Mahwah, NJ, pp. 53-106

[10] Schön D (1983) The reflective practitioner. How professionals think in action. Temple Smith, London

[11] Tsang C, Lau C, Leung Y (2005) Object-Oriented Technology: from Diagram to Code with Visual Paradigm for UML, McGraw-Hill Education, Asia

[12] http://www.compendiuminstitute.org/default.htm

[13] http://teacher.pas.rochester.edu/phy_labs/AppendixE/AppendixE.html

Biographical Data

LEEROY BRONNER is a Research Associate Professor at Morgan State University (MSU) in the department of Industrial Manufacturing and Information Engineering. He has been an instructor at MSU for the past 7 years. Dr. Bronner spent 25 years at IBM Corporation and brings to academia experience in systems, software engineering, modeling, analysis, design, programming and systems implementation.

JERRY-DARYL FLETCHER is a Masters student at Morgan State University's department of Industrial Manufacturing and Information Engineering. He is doing research in the field of Design Rationale for his Masters thesis. He also holds a B.S. degree in electrical engineering from Morgan State University.

Return to Main page